final

Jacob

5/3/2020

```r
library(rvest)
```

```
## Loading required package: xml2
```

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  3.0.0     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter()         masks stats::filter()
## x readr::guess_encoding() masks rvest::guess_encoding()
## x dplyr::lag()            masks stats::lag()
## x purrr::pluck()          masks rvest::pluck()
```

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```r
library(RColorBrewer)
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 0.1.0 --
```

```
## v broom     0.5.5      v rsample   0.0.6
## v dials     0.0.6      v tune      0.1.0
## v infer     0.5.1      v workflows 0.1.1
## v parsnip   0.1.1      v yardstick 0.0.6
## v recipes   0.1.12
```

```
## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x dials::margin()   masks ggplot2::margin()
## x purrr::pluck()    masks rvest::pluck()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
```

```
library(caret) #Confusion matrix, algorithm training
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following objects are masked from 'package:yardstick':
##
##     precision, recall, sensitivity, specificity

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(mice) #Basic data preprocessing - eg. Removing null values(na.omit)etc
```

```
##
## Attaching package: 'mice'

## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
library(ggplot2)#All plots
library(ggcorrplot)#All correlation plot
library(dplyr)#For data manipulation .. eg. selecting numeric variables)
library(openxlsx)#read excel file
library(knitr)
options(stringsAsFactors = FALSE)

songs_2010 <- "dataset-of-10s.csv"
songs_2000 <- "dataset-of-00s.csv"
songs_1990 <- "dataset-of-90s.csv"
songs_1980 <- "dataset-of-80s.csv"
songs_1970 <- "dataset-of-70s.csv"
songs_1960 <- "dataset-of-60s.csv"
```

```r
song_files <- c(songs_2010,songs_2000,songs_1990,songs_1980,songs_1970,songs_1960)
song_df <- NULL
for (song_file in song_files){
  tmp_df <- read.csv(song_file)
  decade <- str_extract(song_file,"\\d{2}")
  tmp_df$decade <- decade
  song_df <- bind_rows(song_df,tmp_df)
}
song_df$decade <- ordered(song_df$decade, levels = c("60","70","80","90","00","10"))
glimpse(song_df)
```

```
## Rows: 41,106
## Columns: 20
## $ track            <chr> "Wild Things", "Surfboard", "Love Someone", "Music...
## $ artist           <chr> "Alessia Cara", "Esquivel!", "Lukas Graham", "Keys...
## $ uri              <chr> "spotify:track:2ZyuwVvV6Z3XJaXIFbspeE", "spotify:t...
## $ danceability     <dbl> 0.741, 0.447, 0.550, 0.502, 0.807, 0.482, 0.533, 0...
## $ energy           <dbl> 0.6260, 0.2470, 0.4150, 0.6480, 0.8870, 0.8730, 0....
## $ key              <int> 1, 5, 9, 0, 1, 0, 0, 2, 7, 8, 1, 2, 5, 0, 1, 2, 8,...
## $ loudness         <dbl> -4.826, -14.661, -6.557, -5.698, -3.892, -3.145, -...
## $ mode             <int> 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1,...
## $ speechiness      <dbl> 0.0886, 0.0346, 0.0520, 0.0527, 0.2750, 0.0853, 0....
## $ acousticness     <dbl> 0.020000, 0.871000, 0.161000, 0.005130, 0.003810, ...
## $ instrumentalness <dbl> 0.00e+00, 8.14e-01, 0.00e+00, 0.00e+00, 0.00e+00, ...
## $ liveness         <dbl> 0.0828, 0.0946, 0.1080, 0.2040, 0.3910, 0.4090, 0....
## $ valence          <dbl> 0.7060, 0.2500, 0.2740, 0.2910, 0.7800, 0.7370, 0....
## $ tempo            <dbl> 108.029, 155.489, 172.065, 91.837, 160.517, 165.08...
## $ duration_ms      <int> 188493, 176880, 205463, 193043, 144244, 214320, 26...
## $ time_signature   <int> 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,...
## $ chorus_hit       <dbl> 41.18681, 33.18083, 44.89147, 29.52521, 24.99199, ...
## $ sections         <int> 10, 9, 9, 7, 8, 12, 14, 10, 11, 9, 10, 13, 12, 8, ...
## $ target           <int> 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,...
## $ decade           <ord> 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10...
```

The code chunk above takes music datasets from the 1960's, 1970's, 1980's, 1990's, 2000's, and 2010's and puts it together to make a larger dataframe containg all songs from all of the datasets. In addition the column "decade" is added containing the decade of each song. This is relevant for the rest of the PROJECT??? since it will allow us to see various aspects of each decade throughout.

```r
song_df <- song_df %>%
  select(-c(uri,key,valence,track))
glimpse(song_df)
```

```
## Rows: 41,106
## Columns: 16
## $ artist           <chr> "Alessia Cara", "Esquivel!", "Lukas Graham", "Keys...
## $ danceability     <dbl> 0.741, 0.447, 0.550, 0.502, 0.807, 0.482, 0.533, 0...
## $ energy           <dbl> 0.6260, 0.2470, 0.4150, 0.6480, 0.8870, 0.8730, 0....
## $ loudness         <dbl> -4.826, -14.661, -6.557, -5.698, -3.892, -3.145, -...
## $ mode             <int> 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1,...
## $ speechiness      <dbl> 0.0886, 0.0346, 0.0520, 0.0527, 0.2750, 0.0853, 0....
## $ acousticness     <dbl> 0.020000, 0.871000, 0.161000, 0.005130, 0.003810, ...
```

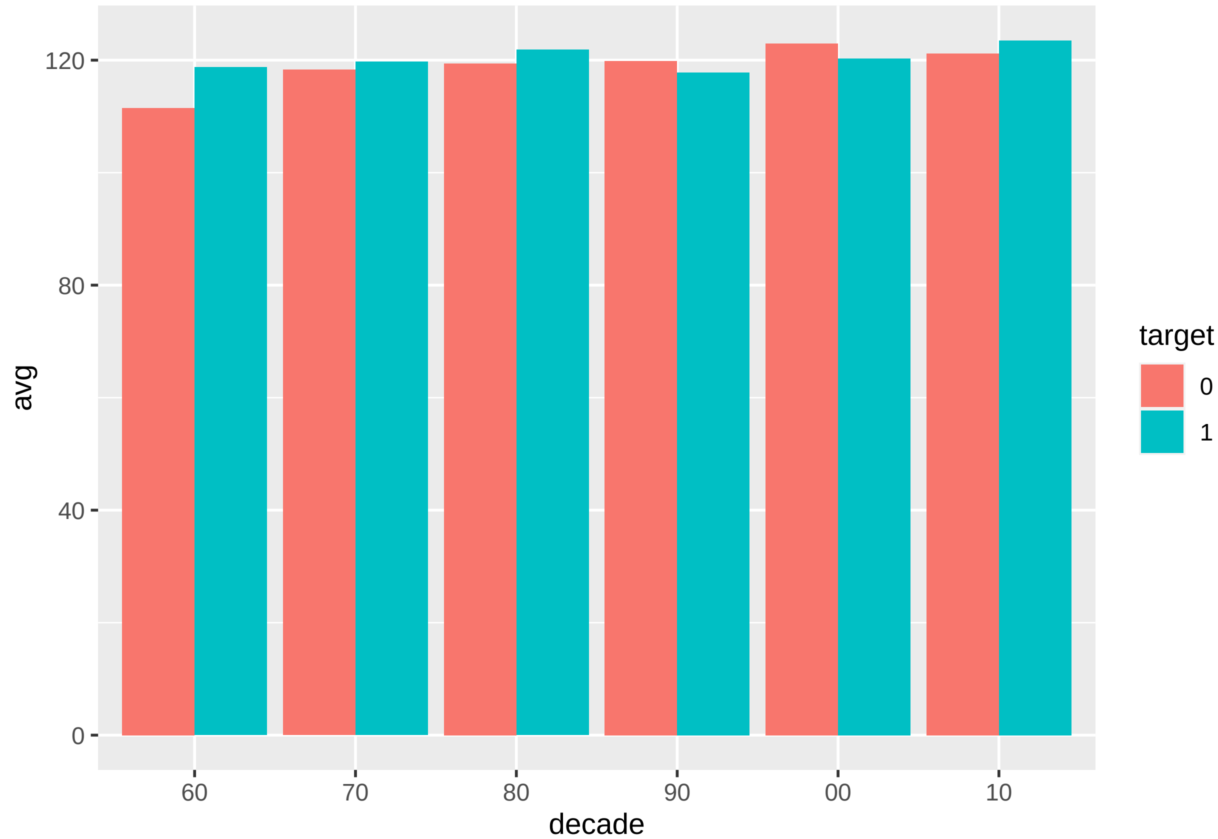
```
## $ instrumentalness <dbl> 0.00e+00, 8.14e-01, 0.00e+00, 0.00e+00, 0.00e+00, ...
## $ liveness         <dbl> 0.0828, 0.0946, 0.1080, 0.2040, 0.3910, 0.4090, 0....
## $ tempo            <dbl> 108.029, 155.489, 172.065, 91.837, 160.517, 165.08...
## $ duration_ms      <int> 188493, 176880, 205463, 193043, 144244, 214320, 26...
## $ time_signature   <int> 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,...
## $ chorus_hit       <dbl> 41.18681, 33.18083, 44.89147, 29.52521, 24.99199, ...
## $ sections         <int> 10, 9, 9, 7, 8, 12, 14, 10, 11, 9, 10, 13, 12, 8, ...
## $ target           <int> 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,...
## $ decade           <ord> 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10...
```

This code chunk is used to get rid of elements of the dataset that are not relevant to this. uri which is the url is not relevant to this as the key song and valence is not relevent ADD STUFF?. track and artist are strings meaning we can't use this in analyzing the data. However, if we wanted to see if there is a different chance for an artist to have a hit given that they have had one previously we could use this. (We would find the first instance of when the artist had a hit and come up with a formula that way)

```r
hits_only <- filter(song_df,target == 1)
glimpse(hits_only)
```

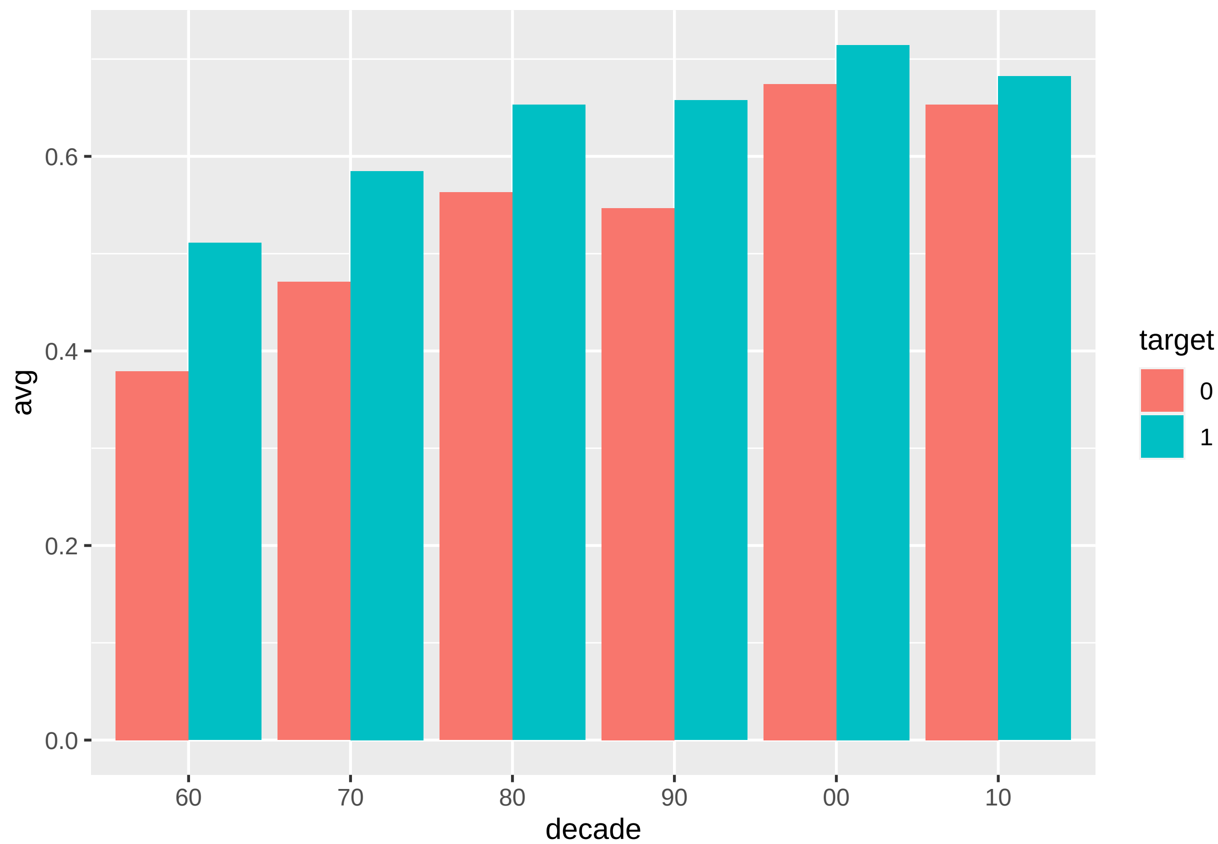
```
## Rows: 20,553
## Columns: 16
## $ artist           <chr> "Alessia Cara", "Lukas Graham", "Zay Hilfigerrr & ...
## $ danceability     <dbl> 0.741, 0.550, 0.807, 0.482, 0.736, 0.387, 0.507, 0...
## $ energy           <dbl> 0.626, 0.415, 0.887, 0.873, 0.522, 0.773, 0.372, 0...
## $ loudness         <dbl> -4.826, -6.557, -3.892, -3.145, -8.020, -5.685, -8...
## $ mode             <int> 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1,...
## $ speechiness      <dbl> 0.0886, 0.0520, 0.2750, 0.0853, 0.1160, 0.1700, 0....
## $ acousticness     <dbl> 0.02000, 0.16100, 0.00381, 0.01110, 0.02990, 0.098...
## $ instrumentalness <dbl> 0.00e+00, 0.00e+00, 0.00e+00, 0.00e+00, 0.00e+00, ...
## $ liveness         <dbl> 0.0828, 0.1080, 0.3910, 0.4090, 0.1080, 0.2090, 0....
## $ tempo            <dbl> 108.029, 172.065, 160.517, 165.084, 97.547, 78.629...
## $ duration_ms      <int> 188493, 205463, 144244, 214320, 200387, 254120, 19...
## $ time_signature   <int> 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,...
## $ chorus_hit       <dbl> 41.18681, 44.89147, 24.99199, 32.17301, 60.21027, ...
## $ sections         <int> 10, 9, 8, 12, 10, 9, 10, 8, 8, 12, 10, 10, 10, 10,...
## $ target           <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
## $ decade           <ord> 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10...
```

```r
plot_avg_by_decade <- function(df,col){

  plotting <- df %>% group_by(decade,target) %>%
    summarise(avg=mean({{col}})) %>% mutate(target=as.factor(target))
  ggplot(plotting) + geom_bar(stat= "identity",aes(x=decade,y = avg,fill=target),position = "dodge")
}
song_df %>% plot_avg_by_decade(tempo)
```



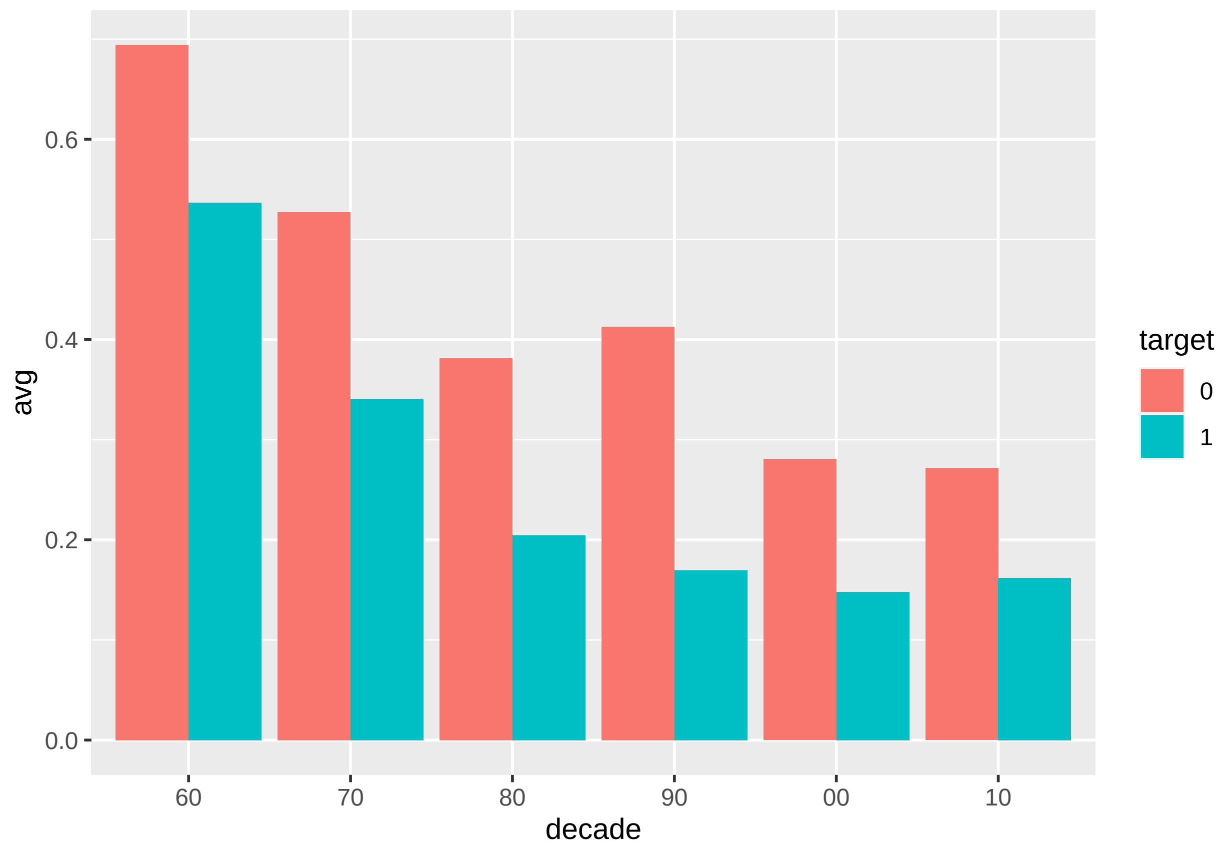
The code chunk above looks to compare average tempo between songs that were hits in each decade which is represented by the red color with all songs that were not hits in each decade which is represented by the color blue. Based off of the graph it does not seem that there is a differrence in the tempo between songs that are hits and songs that are not.

```
plot_avg_by_decade <- function(df,col){

  plotting <- df %>% group_by(decade,target) %>%
    summarise(avg=mean({{col}})) %>% mutate(target=as.factor(target))
  ggplot(plotting) + geom_bar(stat= "identity",aes(x=decade,y = avg,fill=target),position = "dodge")
}
song_df %>% plot_avg_by_decade(energy)
```



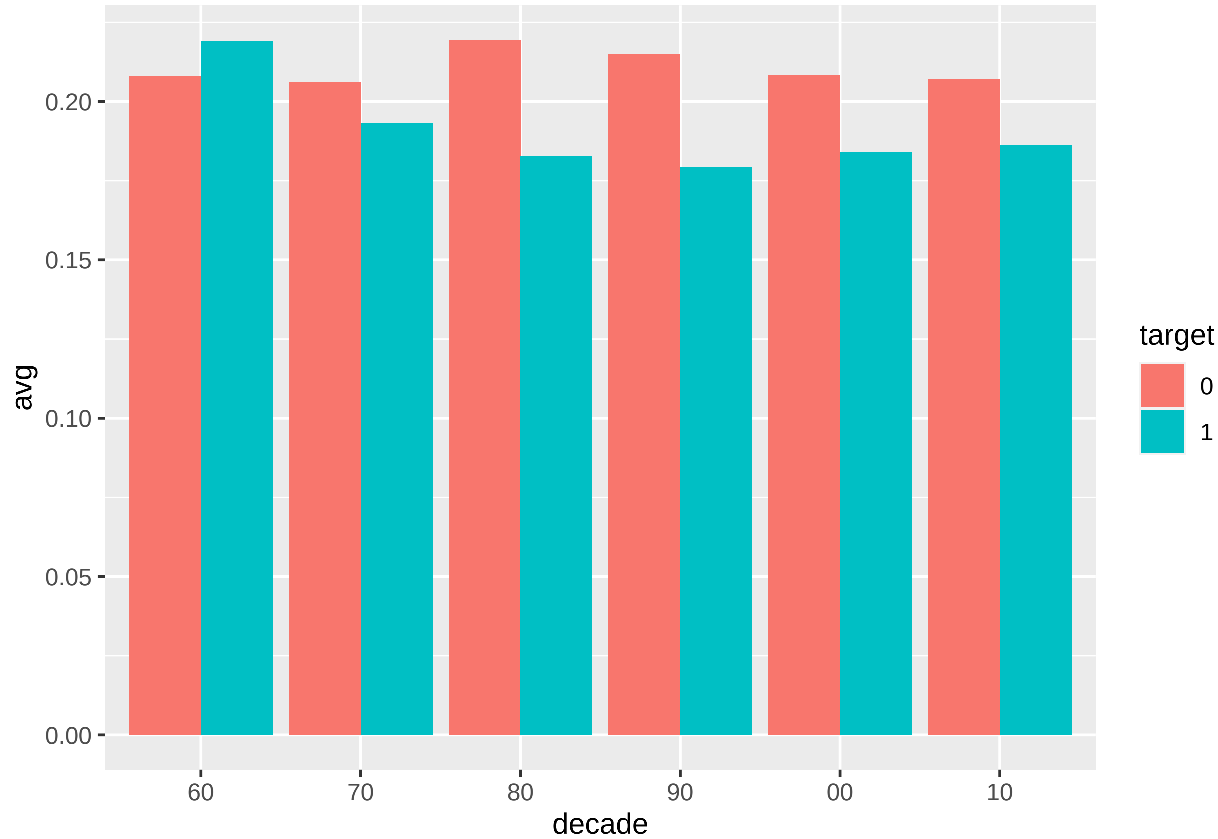
energy

```
plot_avg_by_decade <- function(df,col){

  plotting <- df %>% group_by(decade,target) %>%
    summarise(avg=mean({{col}})) %>% mutate(target=as.factor(target))
  ggplot(plotting) + geom_bar(stat= "identity",aes(x=decade,y = avg,fill=target),position = "dodge")
}
song_df %>% plot_avg_by_decade(acousticness)
```



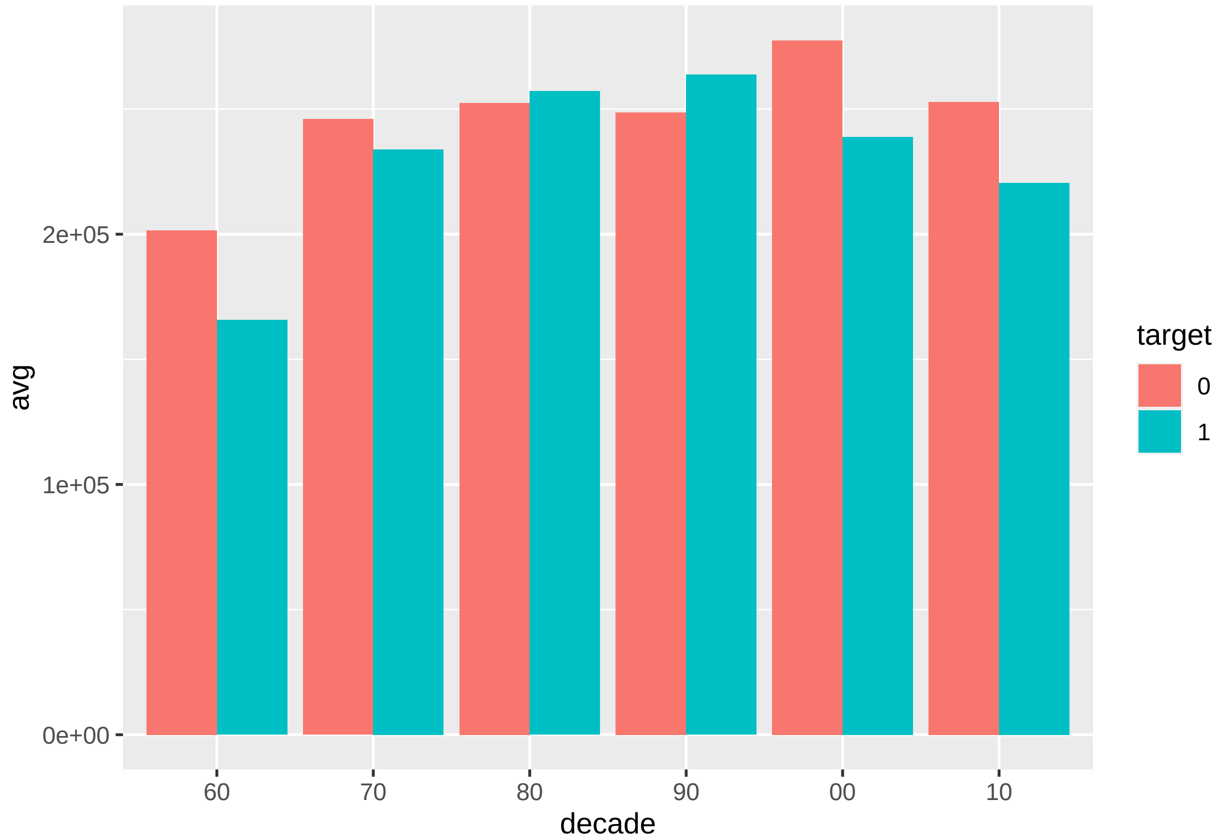
acousticness

```
plot_avg_by_decade <- function(df,col){

  plotting <- df %>% group_by(decade,target) %>%
    summarise(avg=mean({{col}})) %>% mutate(target=as.factor(target))
  ggplot(plotting) + geom_bar(stat= "identity",aes(x=decade,y = avg,fill=target),position = "dodge")
}
song_df %>% plot_avg_by_decade(liveness)
```



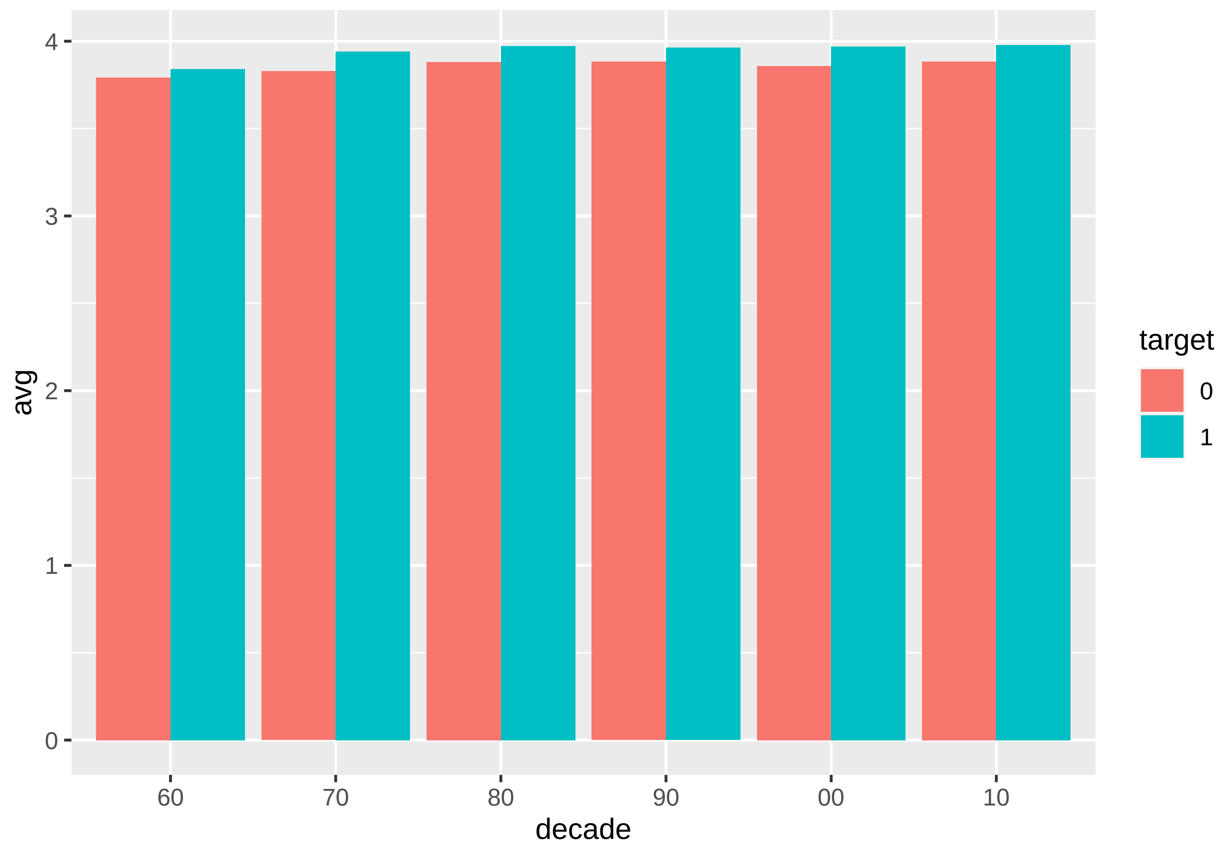
livenss

```
plot_avg_by_decade <- function(df,col){

  plotting <- df %>% group_by(decade,target) %>%
    summarise(avg=mean({{col}})) %>% mutate(target=as.factor(target))
  ggplot(plotting) + geom_bar(stat= "identity",aes(x=decade,y = avg,fill=target),position = "dodge")
}
song_df %>% plot_avg_by_decade(duration_ms)
```



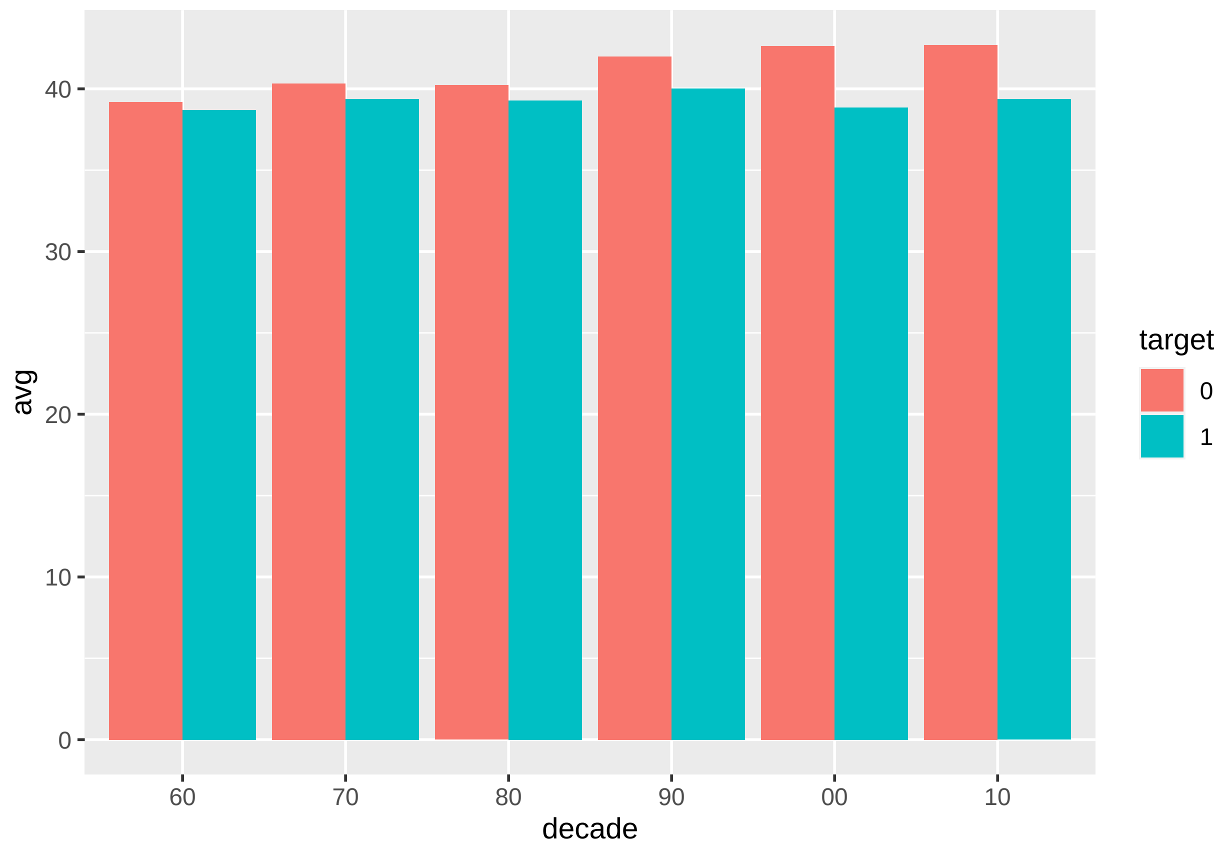
duration

```
plot_avg_by_decade <- function(df,col){

  plotting <- df %>% group_by(decade,target) %>%
    summarise(avg=mean({{col}})) %>% mutate(target=as.factor(target))
  ggplot(plotting) + geom_bar(stat= "identity",aes(x=decade,y = avg,fill=target),position = "dodge")
}
song_df %>% plot_avg_by_decade(time_signature)
```



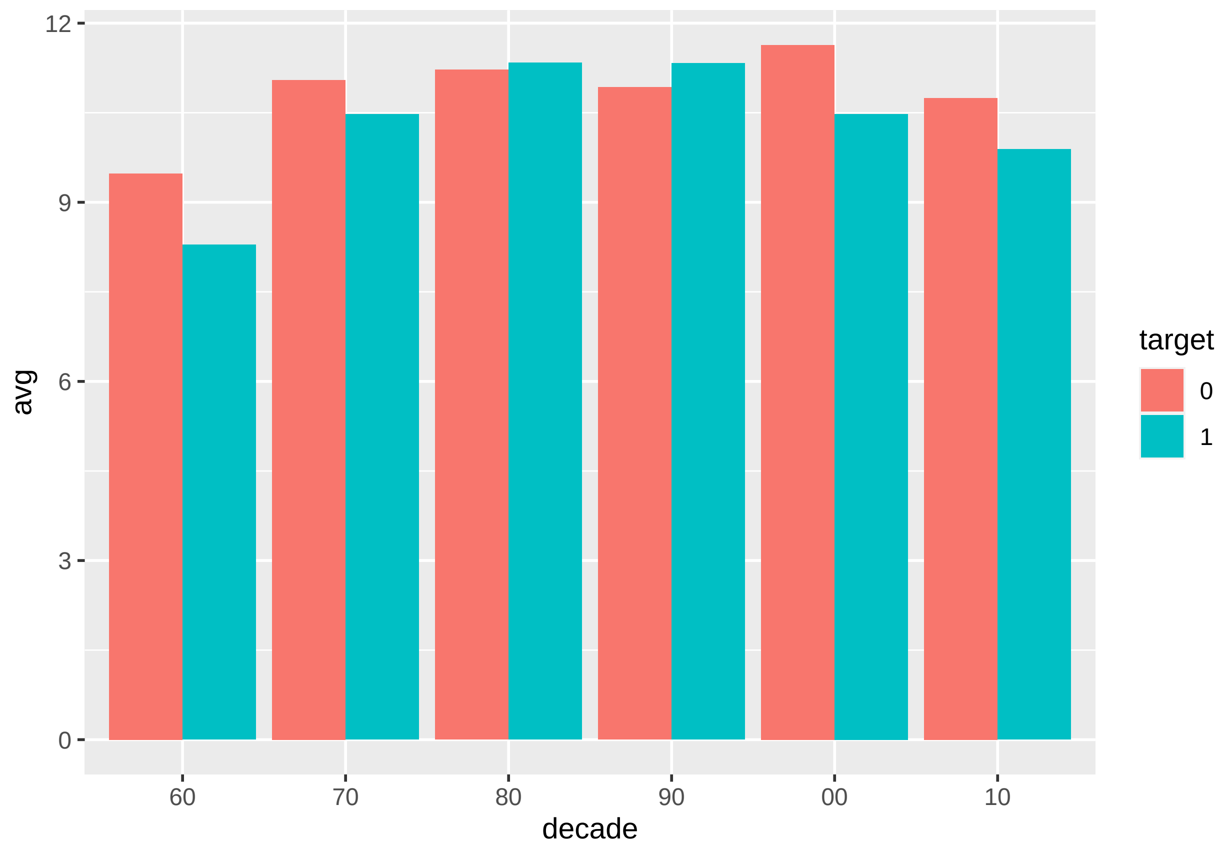
time signature

```
plot_avg_by_decade <- function(df,col){

  plotting <- df %>% group_by(decade,target) %>%
    summarise(avg=mean({{col}})) %>% mutate(target=as.factor(target))
  ggplot(plotting) + geom_bar(stat= "identity",aes(x=decade,y = avg,fill=target),position = "dodge")
}
song_df %>% plot_avg_by_decade(chorus_hit)
```



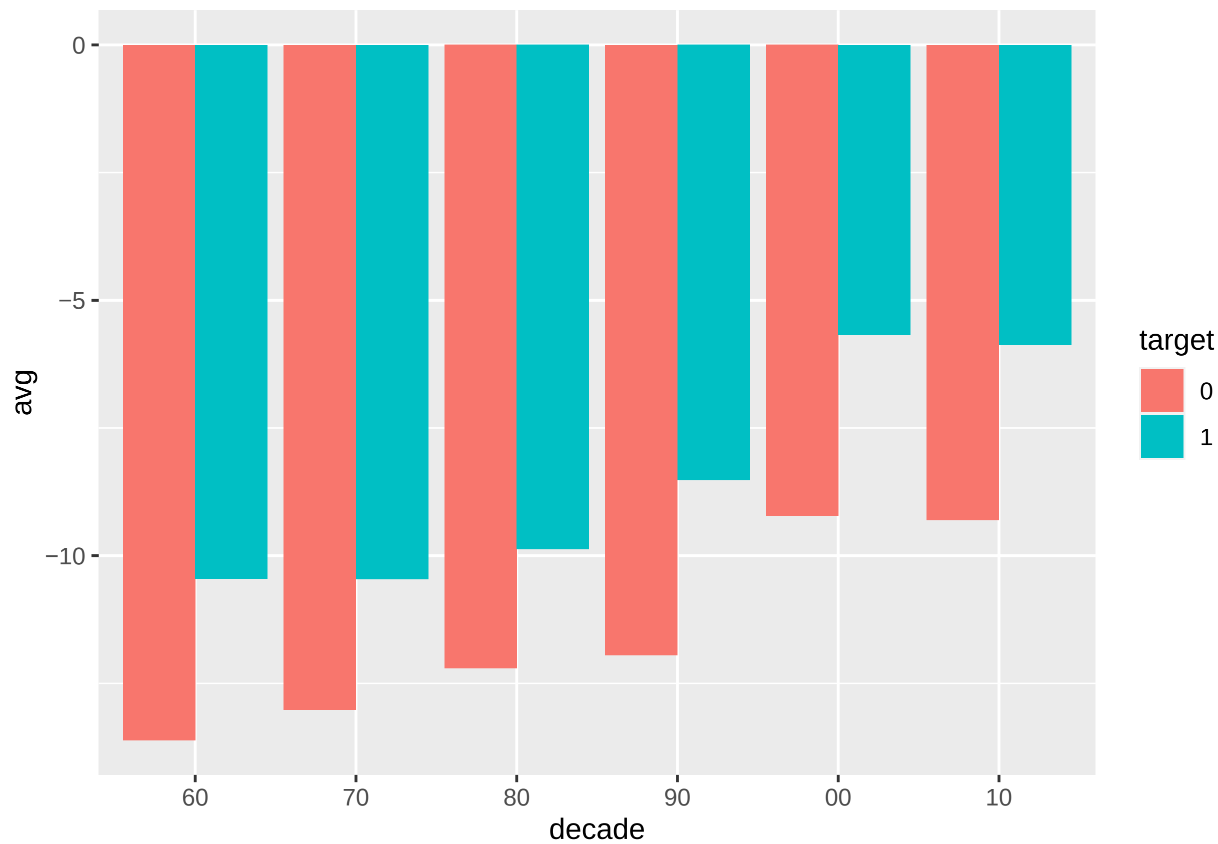
chorus

```
plot_avg_by_decade <- function(df,col){

  plotting <- df %>% group_by(decade,target) %>%
    summarise(avg=mean({{col}})) %>% mutate(target=as.factor(target))
  ggplot(plotting) + geom_bar(stat= "identity",aes(x=decade,y = avg,fill=target),position = "dodge")
}
song_df %>% plot_avg_by_decade(sections)
```



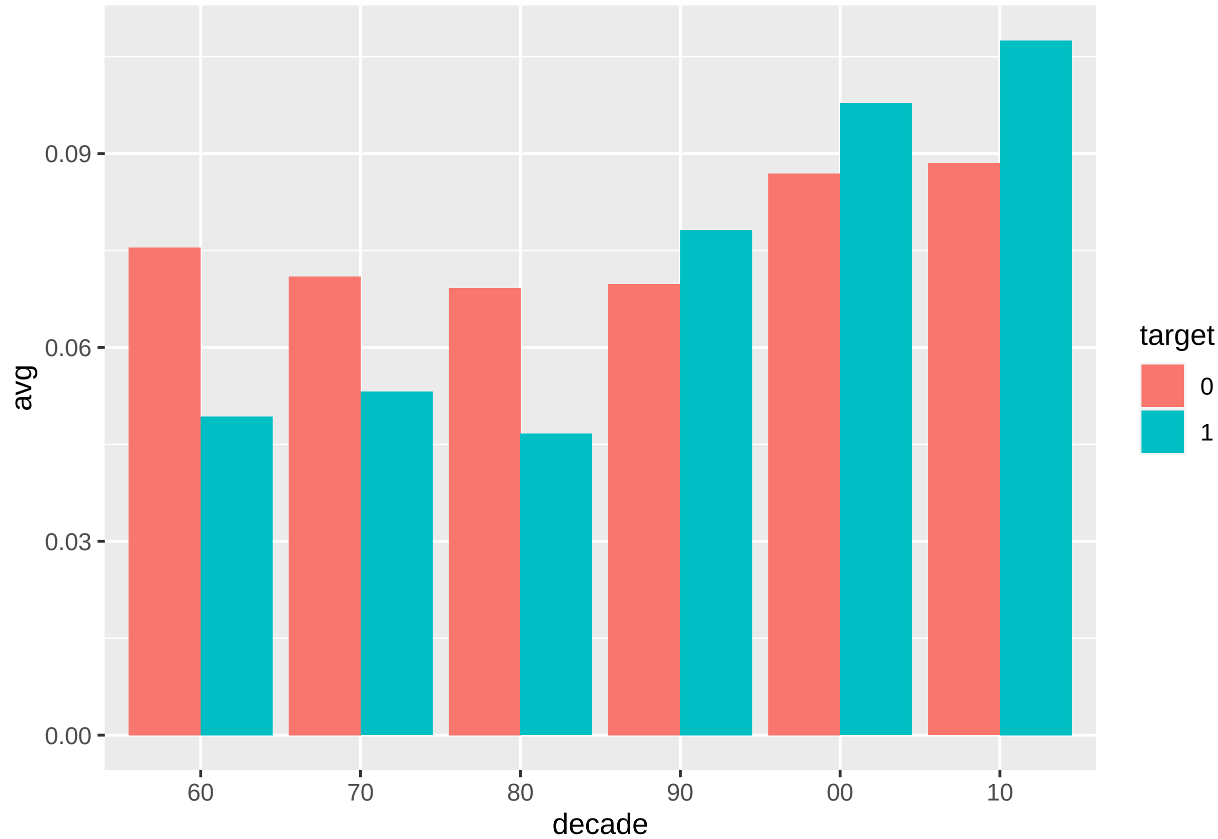
sections

```
plot_avg_by_decade <- function(df,col){

  plotting <- df %>% group_by(decade,target) %>%
    summarise(avg=mean({{col}})) %>% mutate(target=as.factor(target))
  ggplot(plotting) + geom_bar(stat= "identity",aes(x=decade,y = avg,fill=target),position = "dodge")
}
song_df %>% plot_avg_by_decade(loudness)
```



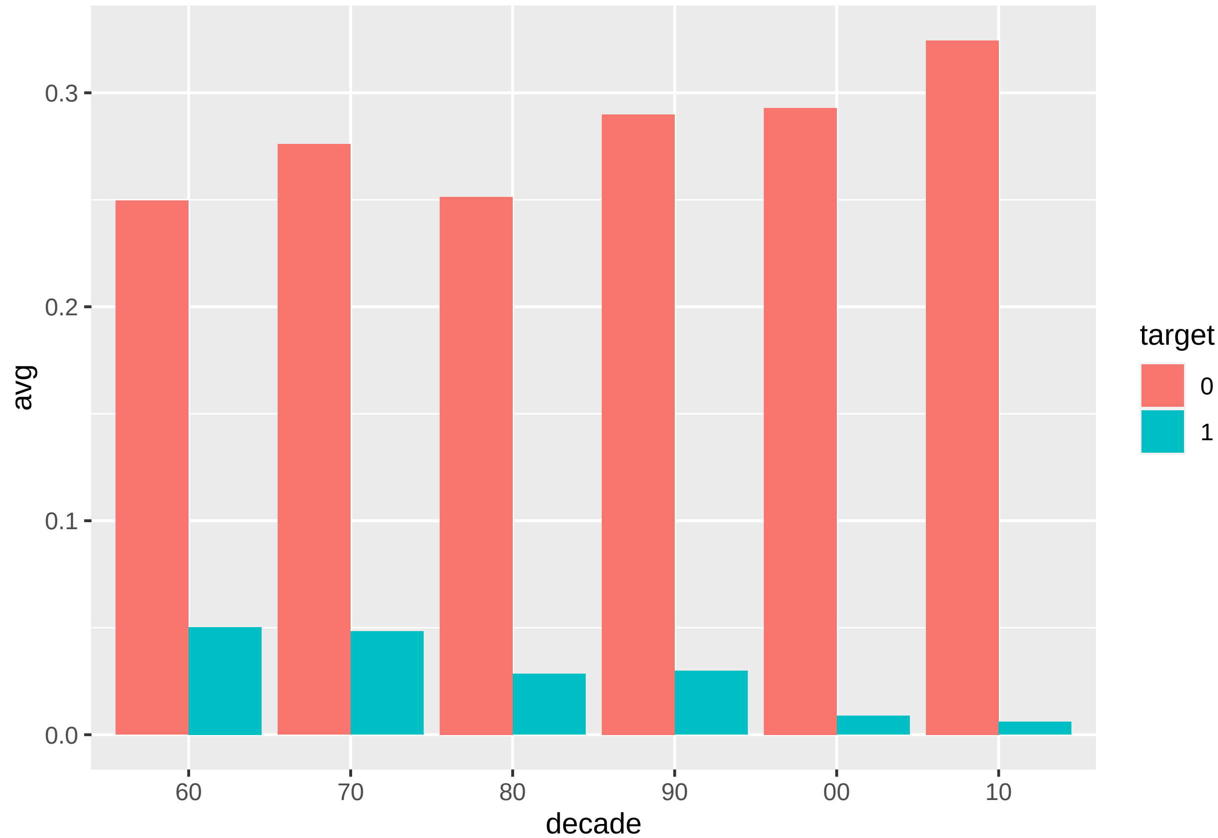
loudness

```
plot_avg_by_decade <- function(df,col){

  plotting <- df %>% group_by(decade,target) %>%
    summarise(avg=mean({{col}})) %>% mutate(target=as.factor(target))
  ggplot(plotting) + geom_bar(stat= "identity",aes(x=decade,y = avg,fill=target),position = "dodge")
}
song_df %>% plot_avg_by_decade(speechiness)
```



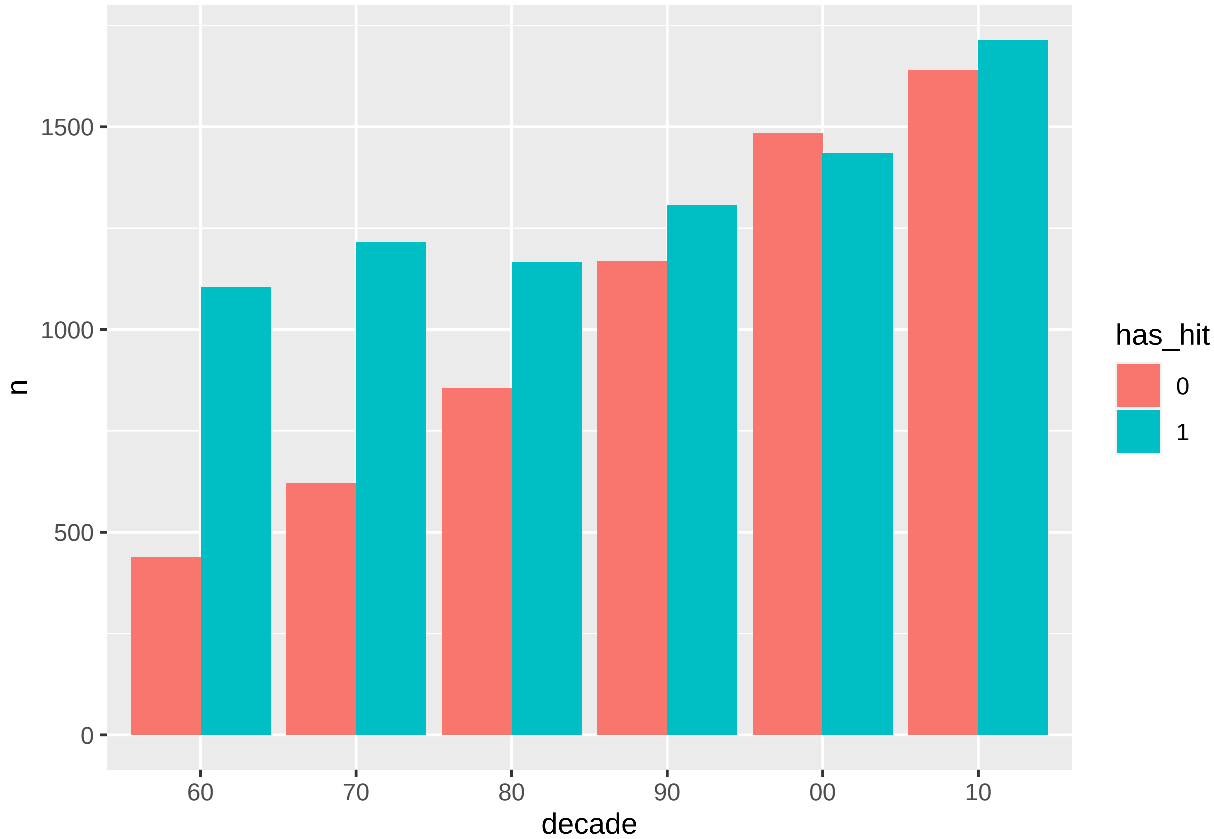
speechness

```
plot_avg_by_decade <- function(df,col){

  plotting <- df %>% group_by(decade,target) %>%
    summarise(avg=mean({{col}})) %>% mutate(target=as.factor(target))
  ggplot(plotting) + geom_bar(stat= "identity",aes(x=decade,y = avg,fill=target),position = "dodge")
}
song_df %>% plot_avg_by_decade(instrumentalness)
```



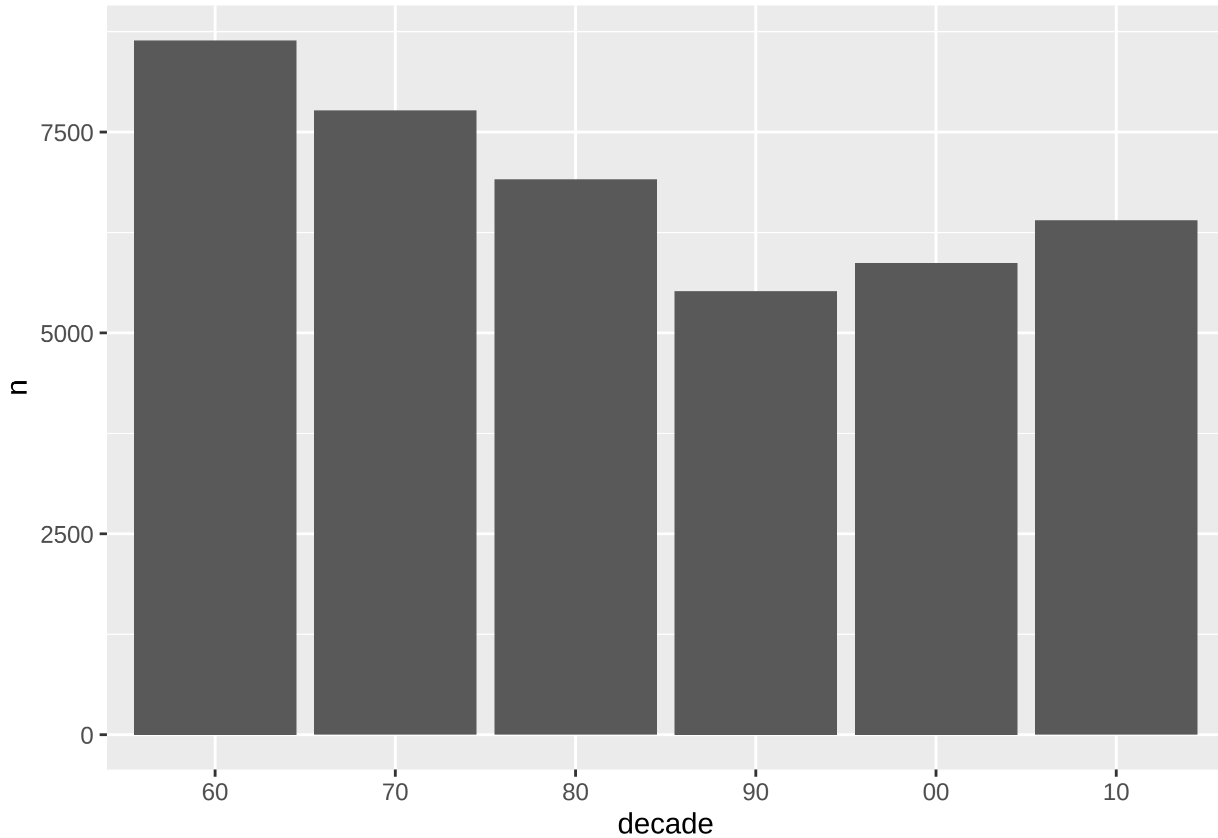
intstrumentalness

```
song_df %>%
  group_by(decade,artist) %>%
  summarise(has_hit=max(target))%>%
  mutate(has_hit=as.factor(has_hit)) %>%
  group_by(decade,has_hit)%>%
  count -> hit_amount
ggplot(hit_amount) + geom_bar(stat="identity", aes(x=decade,y=n,fill=has_hit),position = "dodge")
```



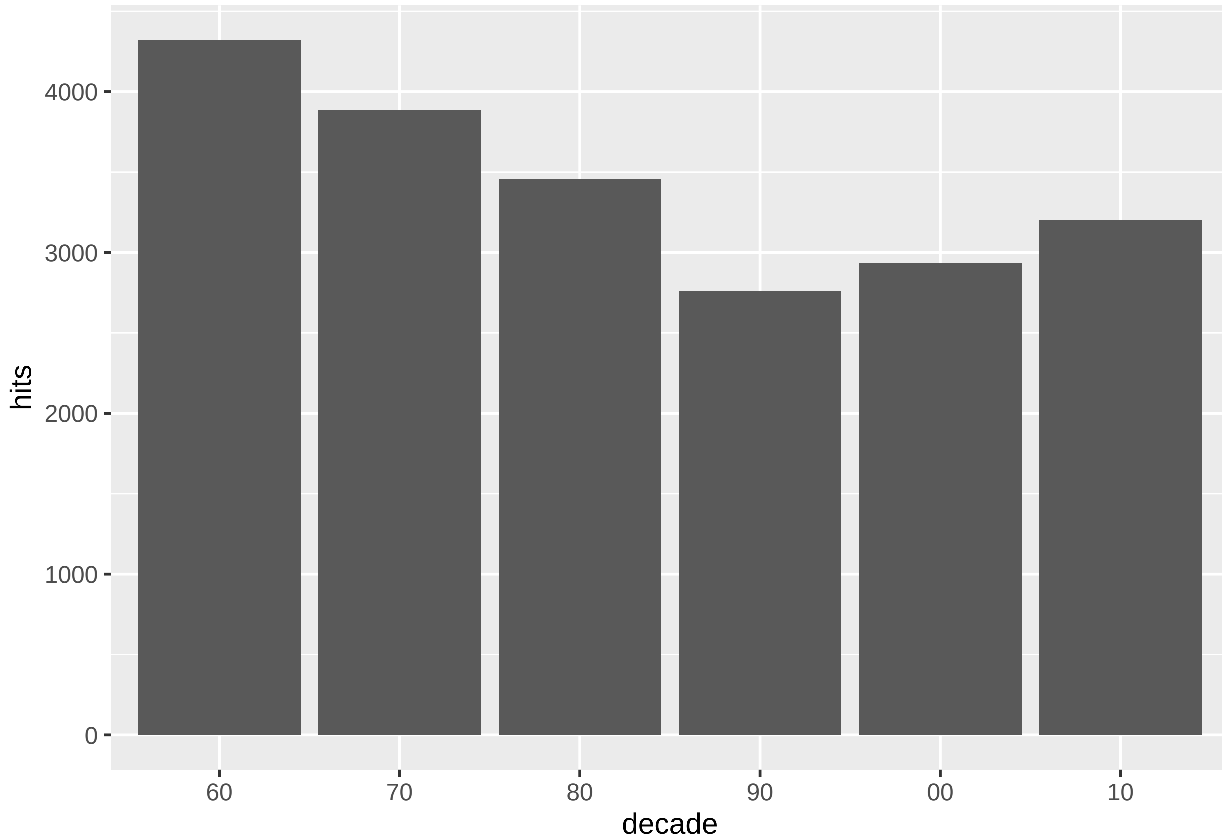
how many artists have hits

```
song_df %>%
  group_by(decade) %>%
  count -> freq
  ggplot(freq) + geom_bar(stat= "identity",aes(x=decade,y = n),position = "dodge")
```



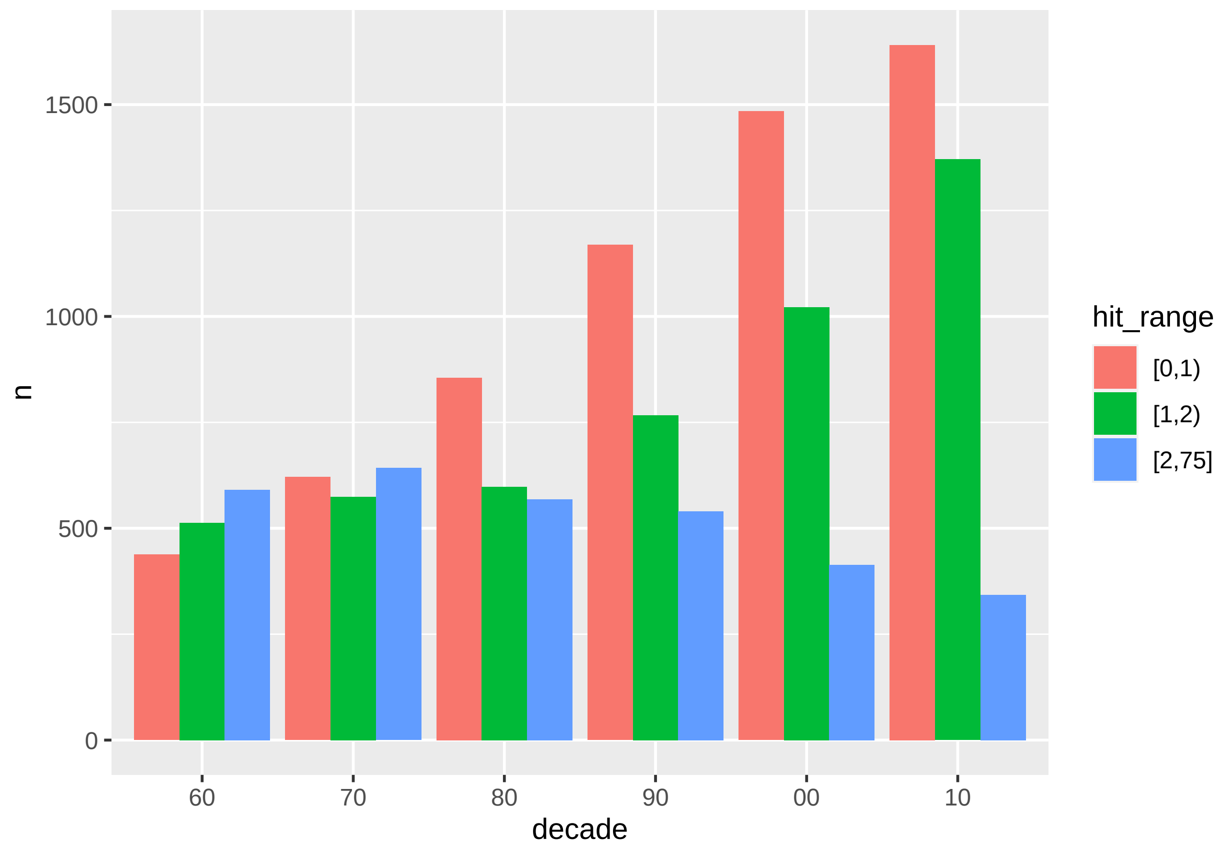
#how many songs in each decade

```
plot_total_hits_by_decade <- function(df,col){

  plotting <- df %>% group_by(decade) %>%
    summarise(hits=sum({{col}}))
  ggplot(plotting) + geom_bar(stat= "identity",aes(x=decade,y = hits))
}
song_df %>% plot_total_hits_by_decade(target)
```



hits per decade This graph looks at the total amount of hits per decade. This is relevant because it shows that there was more variance in terms of how many different songs were hits. Leads to questions of how many different artists had hits to see if popularity of artists determined the amount of hits.

```
song_df %>%
  group_by(decade,artist) %>%
  summarise(num_hits=sum(target))%>%
  mutate(hit_range=cut(num_hits,c(0,1,2,75),right=FALSE,include.lowest = TRUE))%>%
  group_by(decade,hit_range)%>%
  count -> tot_hit
ggplot(tot_hit) + geom_bar(stat="identity", aes(x=decade,y=n,fill=hit_range),position = "dodge")
```



#Compares artists who had 0 hits with 1 hit and more than 1 hit

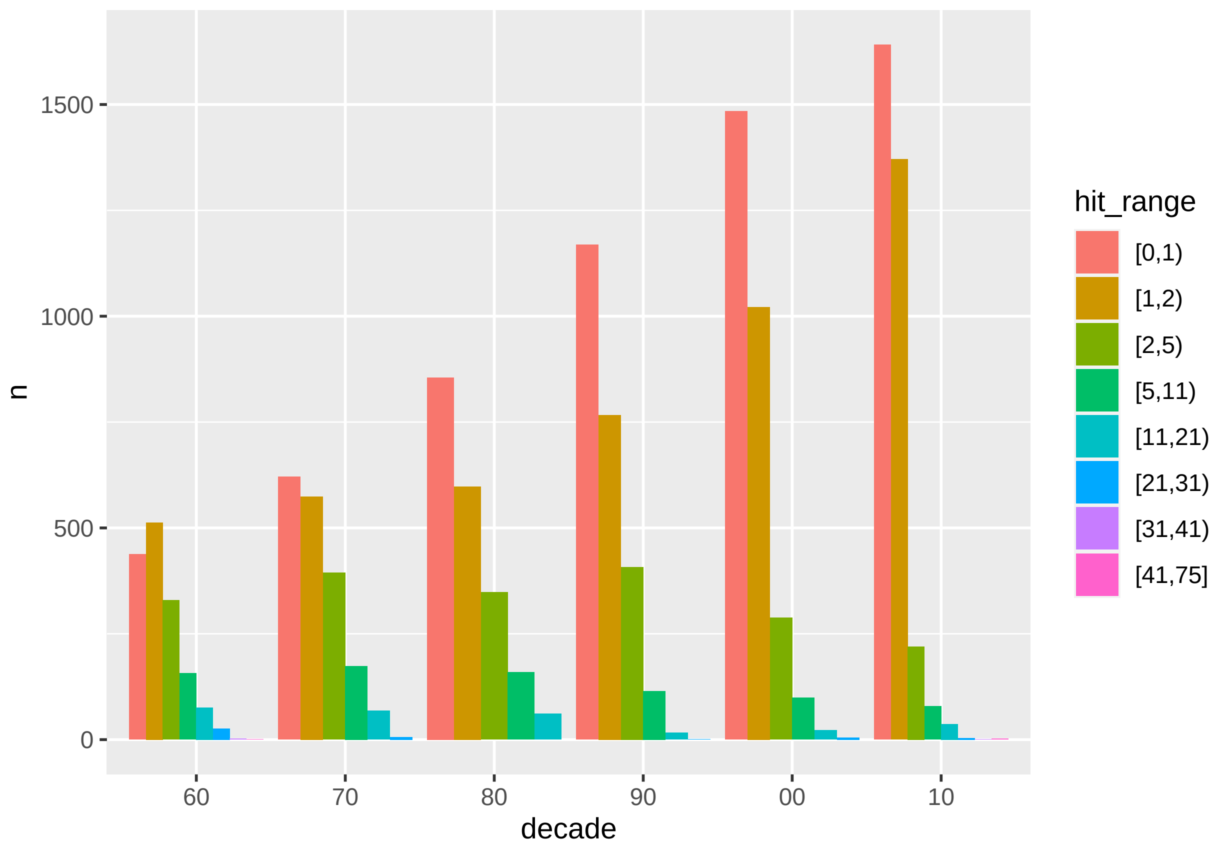
```
song_df %>%
  group_by(decade,artist) %>%
  summarise(num_hits=sum(target))%>%
  mutate(hit_range=cut(num_hits,c(0,1,2,75),right=FALSE,include.lowest = TRUE))%>%
  group_by(decade,hit_range)%>%
  count -> tot_hit
summarise(tot_hit,n)
```

```
## # A tibble: 18 x 3
## # Groups:   decade [6]
##    decade hit_range     n
##    <ord>  <fct>     <int>
##  1 60     [0,1)       438
##  2 60     [1,2)       513
##  3 60     [2,75]      591
##  4 70     [0,1)       621
##  5 70     [1,2)       574
##  6 70     [2,75]      642
##  7 80     [0,1)       855
##  8 80     [1,2)       598
##  9 80     [2,75]      568
## 10 90     [0,1)      1169
## 11 90     [1,2)       767
## 12 90     [2,75]      540
```

```
## 13 00      [0,1)       1484
## 14 00      [1,2)       1022
## 15 00      [2,75]       414
## 16 10      [0,1)       1641
## 17 10      [1,2)       1371
## 18 10      [2,75]       343
```

#compares growth between ranges of hits. seeing how many hits an artist needs before they reach a certain amount of hits –> Stardom? break up this data more to get better precentages for determining the tipping point between "guaranteed hit"

```r
song_df %>%
  group_by(decade,artist) %>%
  summarise(num_hits=sum(target))%>%
  mutate(hit_range=cut(num_hits,c(0,1,2,5,11,21,31,41,75),right=FALSE,include.lowest = TRUE))%>%
  group_by(decade,hit_range)%>%
  count -> tot_hit
ggplot(tot_hit) + geom_bar(stat="identity", aes(x=decade,y=n,fill=hit_range),position = "dodge")
```



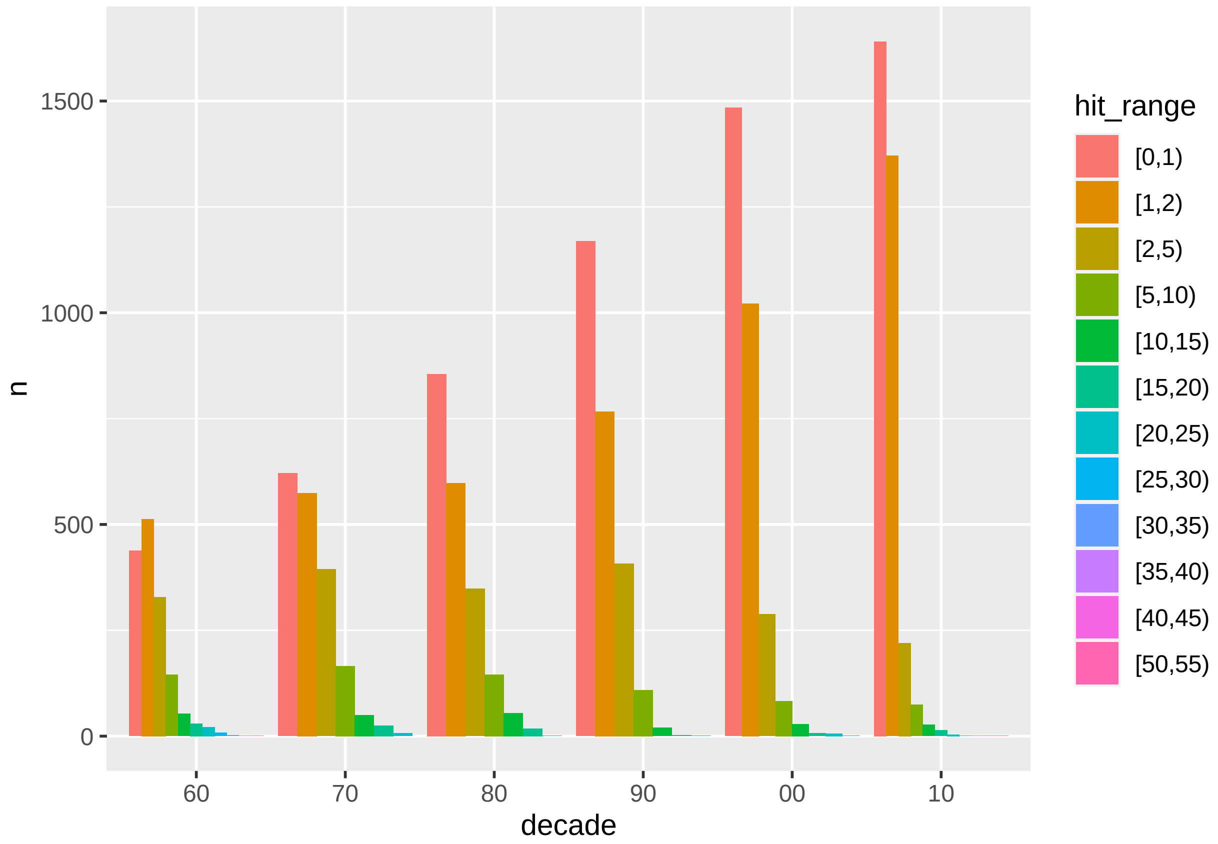

```r
# %of hits from artists who already have one vs % of hits from those who don't(total artists with hits 

#artist likelihood of having a previous hit affect liklihood of another song becoming a hit (when does
```

#shows how many artists had how many hits on ranges in each decade.

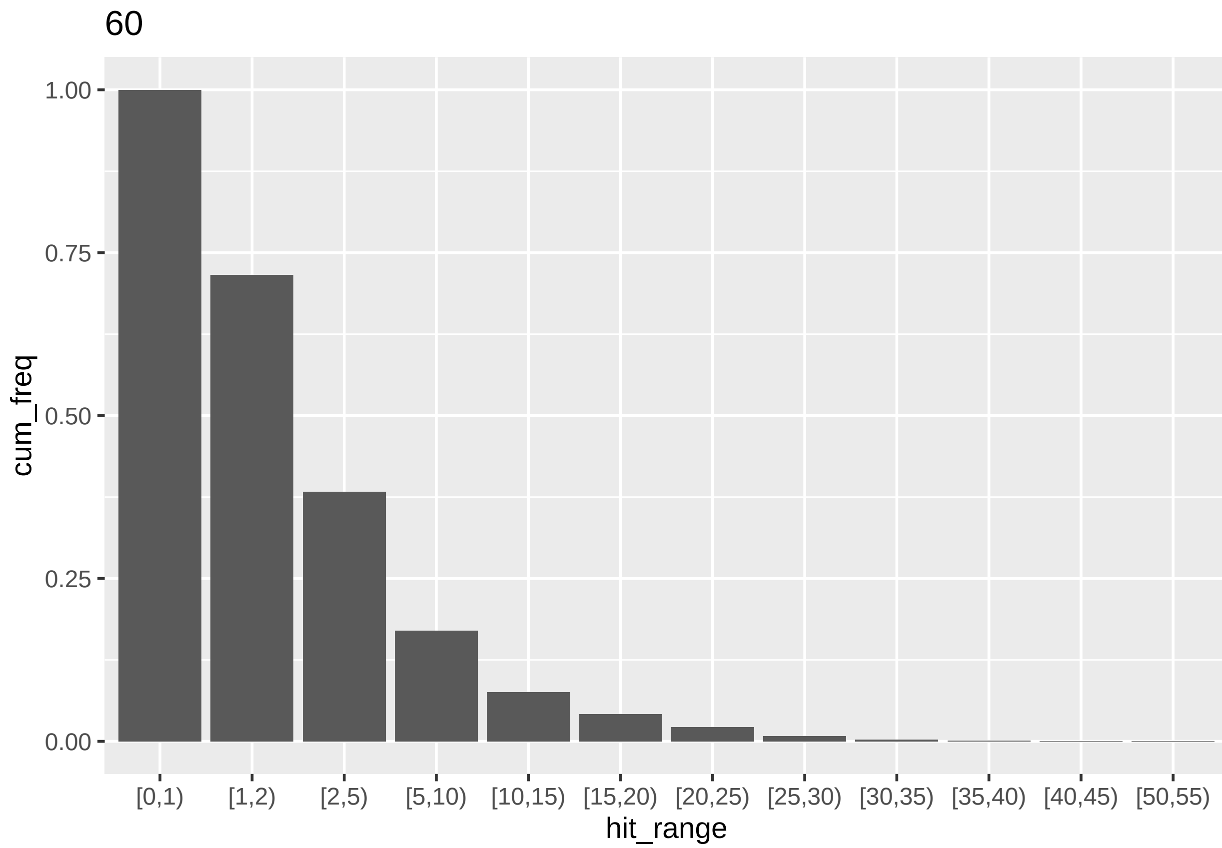
```
song_df %>%
  group_by(decade,artist) %>%
  summarise(num_hits=sum(target))%>%
  mutate(hit_range=cut(num_hits,c(0,1,2,5,10,15,20,25,30,35,40,45,50,55,60,65,70,75),right=FALSE,includ
  group_by(decade,hit_range)%>%
  count -> tot_hit
summarise(tot_hit,n)
```

```
## # A tibble: 51 x 3
## # Groups:   decade [6]
##    decade hit_range     n
##    <ord>  <fct>     <int>
##  1 60     [0,1)       438
##  2 60     [1,2)       513
##  3 60     [2,5)       329
##  4 60     [5,10)      145
##  5 60     [10,15)      53
##  6 60     [15,20)      30
##  7 60     [20,25)      22
##  8 60     [25,30)       8
##  9 60     [30,35)       2
## 10 60     [35,40)       1
## # ... with 41 more rows
```

```
ggplot(tot_hit) + geom_bar(stat="identity", aes(x=decade,y=n,fill=hit_range),position = "dodge")
```

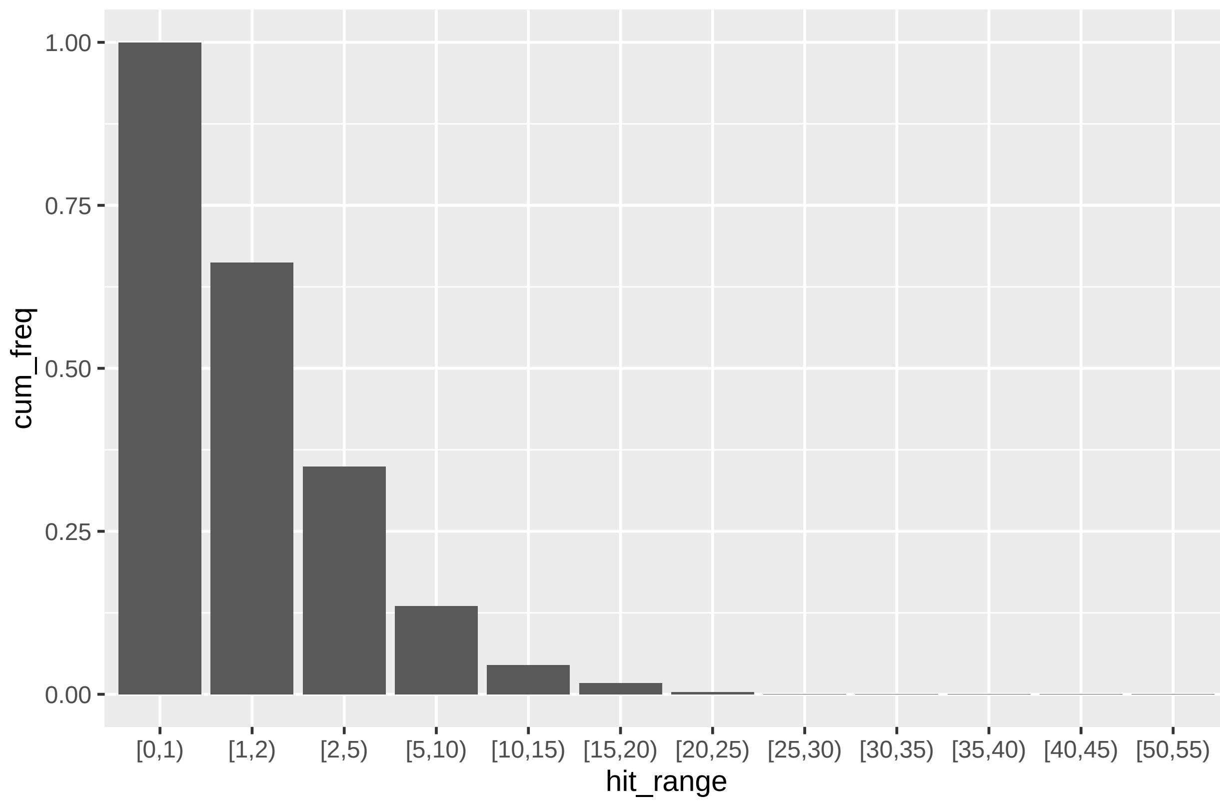


```
song_df %>%
  group_by(decade,artist) %>%
  summarise(num_hits=sum(target))%>%
  mutate(hit_range=cut(num_hits,c(0,1,2,5,10,15,20,25,30,35,40,45,50,55,60,65,70,75),right=FALSE,includ
  group_by(decade,hit_range)%>%
  summarise(n=n()) %>%
  ungroup() -> mid_df

hit_ranges <- mid_df %>% ungroup() %>%
  distinct(hit_range) %>%
  mutate(join_var = 1)

decades <- mid_df %>%
  distinct(decade) %>%
  arrange(decade) %>% mutate(join_var = 1)
all_combo <- hit_ranges %>%
  inner_join(decades,by = "join_var") %>%
  select(-join_var)

all_combo %>%
  left_join(mid_df,by = c("decade","hit_range")) %>%
  replace_na(list(n=0)) %>%
  group_by(decade) %>%
  arrange(hit_range) %>%
  mutate(freq = n/sum(n),cum_freq = rev(cumsum(rev(freq)))) %>%
  ungroup() -> per_df


decades <- per_df %>%
  distinct(decade) %>%
  arrange(decade) %>%
  unlist() %>%
  unname()

plot_lst = list()
for (dec in decades) {
  plot_lst [[dec]] <- ggplot(per_df %>% filter(decade == dec)) + geom_bar(aes(x = hit_range,y=cum_freq)
}
plot_lst[["60"]]
```

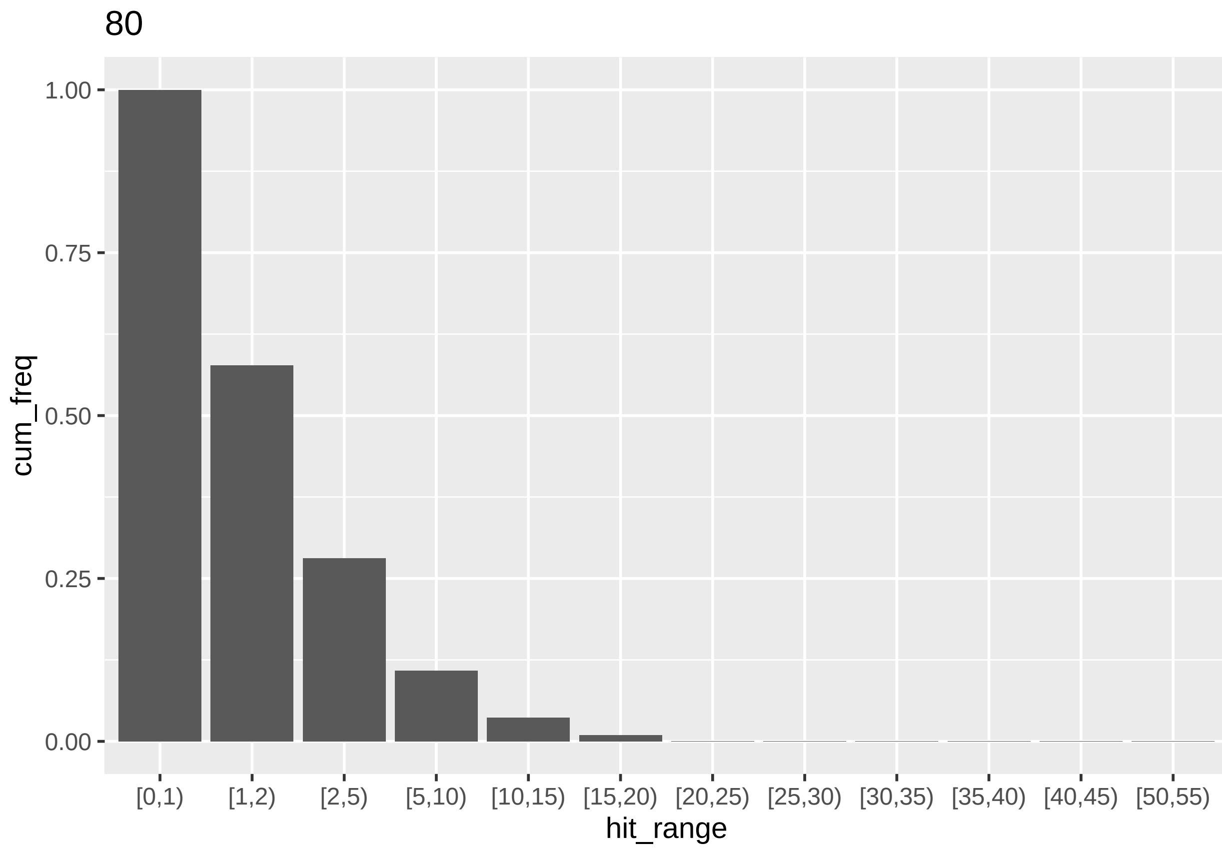

60
1.00
0.75
0.50
0.25
0.00
cum_freq
[0,1)
[1,2)
[2,5)
[5,10)
[10,15)
[15,20)
[20,25)
[25,30)
[30,35)
[35,40)
[40,45)
[50,55)
hit_range

```
plot_lst[["70"]]
```

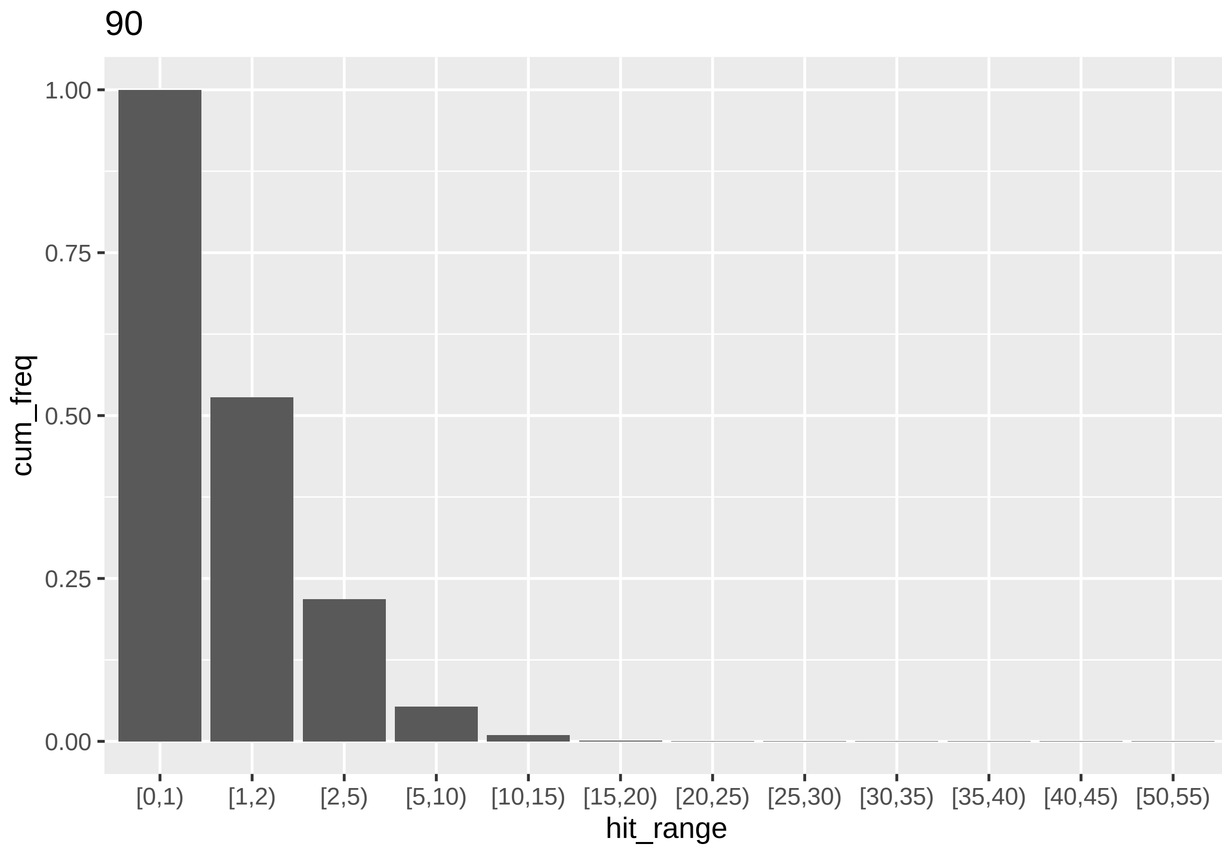
70




```
plot_lst[["80"]]
```

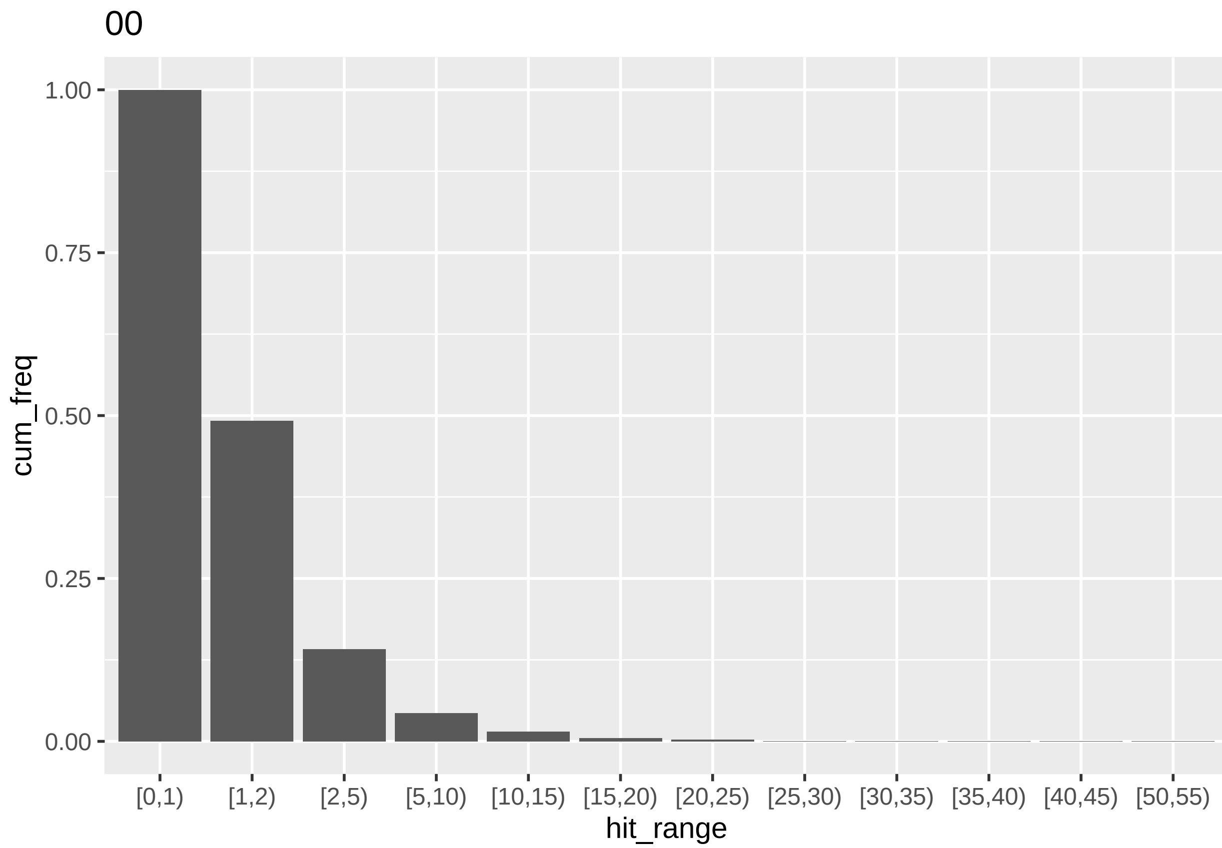

80
1.00
0.75
0.50
0.25
0.00
cum_freq
[0,1)
[1,2)
[2,5)
[5,10)
[10,15)
[15,20)
[20,25)
[25,30)
[30,35)
[35,40)
[40,45)
[50,55)
hit_range

```
plot_lst[["90"]]
```

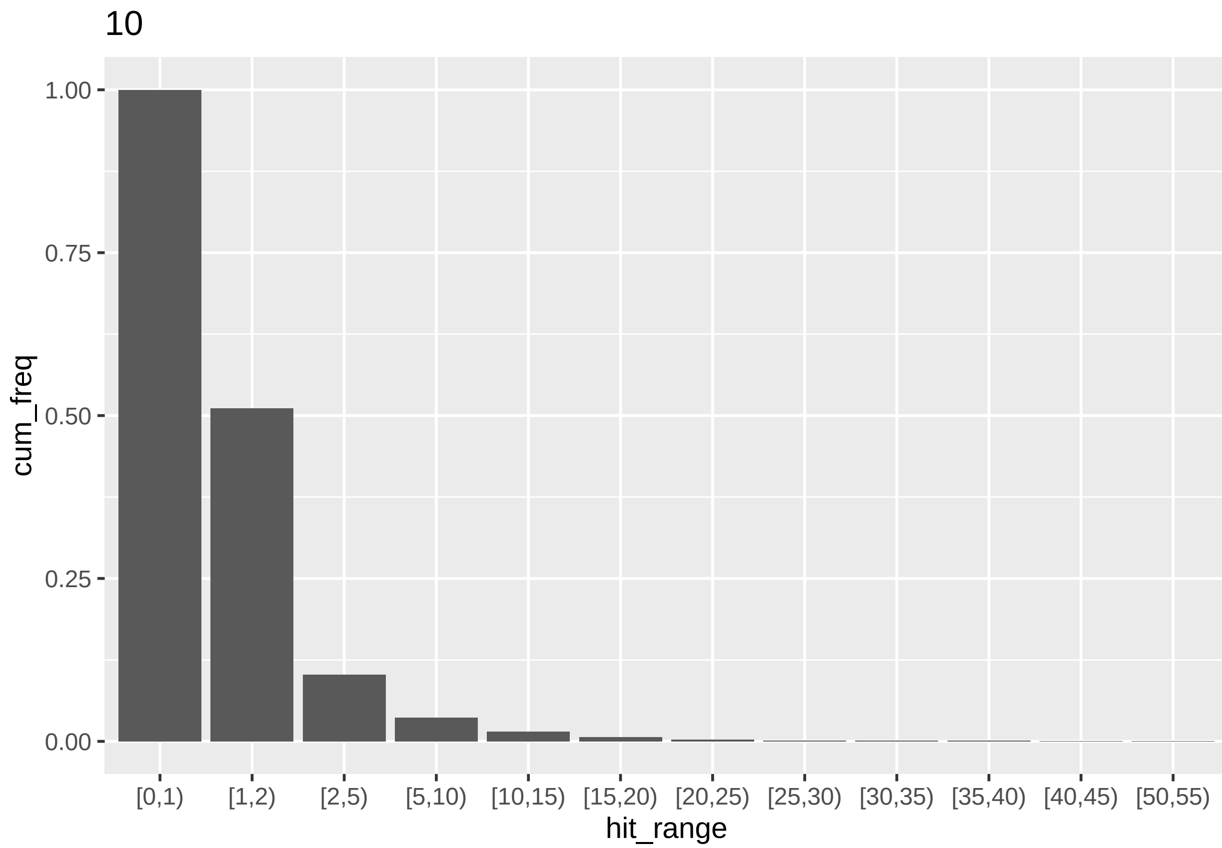


90
1.00
0.75
0.50
0.25
0.00
cum_freq
[0,1)
[1,2)
[2,5)
[5,10)
[10,15)
[15,20)
[20,25)
[25,30)
[30,35)
[35,40)
[40,45)
[50,55)
hit_range


```
plot_lst[["00"]]
```


00
1.00
0.75
0.50
0.25
0.00
cum_freq
[0,1)
[1,2)
[2,5)
[5,10)
[10,15)
[15,20)
[20,25)
[25,30)
[30,35)
[35,40)
[40,45)
[50,55)
hit_range

```
plot_lst[["10"]]
```



#total hits in each decade will divide each entity going up showing percentages

#if artist already has a hit go in 'b' else 'a'

```
song_df %>%
  select(-c(artist,mode,sections,time_signature)) %>%
  mutate(target = as.factor(target))-> update_song_df
data_split <- initial_split(update_song_df,prop = 0.8)
data_recipe <- data_split %>% training() %>% recipe(target~.) %>%
  step_corr(all_numeric()) %>%
  step_center(all_numeric(),-all_outcomes()) %>%
  step_scale(all_numeric(),-all_outcomes()) %>%
  prep()

data_testing <- data_recipe %>%
  bake(testing(data_split))
data_training <- juice(data_recipe)
```

data prep

```
r_forest <- rand_forest(trees = 100, mode = "classification") %>%
  set_engine("randomForest") %>%
  fit(target~., data = data_training)
```

random forest

```r
r_forest %>%
  predict(data_testing) %>%
  bind_cols(data_testing) -> predictions_r_forest
  predictions_r_forest %>%
    metrics(truth = target,estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.809
## 2 kap      binary         0.618
```

random forest new chunk to prevent the constant running of previous chunk based off of our predictions we have an 80% accuracy on unseen data

```r
log_reg <- logistic_reg(mode = "classification") %>%
  set_engine("glm") %>%
  fit(target~., data = data_training)
log_reg %>%
  predict(data_testing) %>%
  bind_cols(data_testing) -> predictions_log_reg
  predictions_log_reg %>%
    metrics(truth = target,estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.744
## 2 kap      binary         0.488
```

logistic regression 73% accuracy

Multiple Logistic Regression

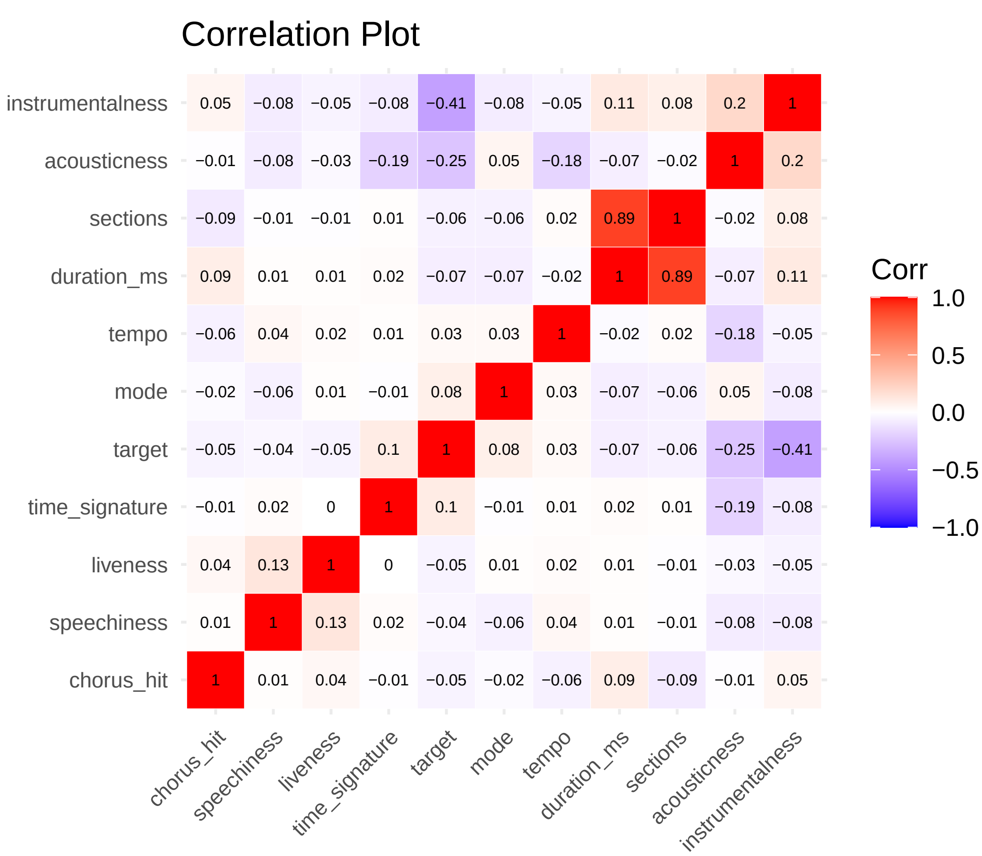
```r
musiclog = song_df[,-c(1,2,3,4)]#removing the  X ,track,artist,uri as these cannot be modelled

#summary of dataset
summary(musiclog)
```

```
##       mode         speechiness       acousticness    instrumentalness
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.0000   1st Qu.:0.03370   1st Qu.:0.0394   1st Qu.:0.00000
##  Median :1.0000   Median :0.04340   Median :0.2580   Median :0.00012
##  Mean   :0.6934   Mean   :0.07296   Mean   :0.3642   Mean   :0.15442
##  3rd Qu.:1.0000   3rd Qu.:0.06980   3rd Qu.:0.6760   3rd Qu.:0.06125
##  Max.   :1.0000   Max.   :0.96000   Max.   :0.9960   Max.   :1.00000
##     liveness          tempo        duration_ms      time_signature
##  Min.   :0.0130   Min.   :  0.0   Min.   :  15168   Min.   :0.000
##  1st Qu.:0.0940   1st Qu.: 97.4   1st Qu.: 172928   1st Qu.:4.000
```

```
## Median :0.1320   Median :117.6   Median : 217907   Median :4.000
## Mean   :0.2015   Mean   :119.3   Mean   : 234878   Mean   :3.894
## 3rd Qu.:0.2610   3rd Qu.:136.5   3rd Qu.: 266773   3rd Qu.:4.000
## Max.   :0.9990   Max.   :241.4   Max.   :4170227   Max.   :5.000
##   chorus_hit        sections          target     decade
## Min.   :  0.00   Min.   :  0.00   Min.   :0.0   60:8642
## 1st Qu.: 27.60   1st Qu.:  8.00   1st Qu.:0.0   70:7766
## Median : 35.85   Median : 10.00   Median :0.5   80:6908
## Mean   : 40.11   Mean   : 10.48   Mean   :0.5   90:5520
## 3rd Qu.: 47.63   3rd Qu.: 12.00   3rd Qu.:1.0   00:5872
## Max.   :433.18   Max.   :169.00   Max.   :1.0   10:6398
```

Correlation Analysis

```
musiclognum = musiclog%>% select_if(is.numeric)#selecting only the numeric variables from student for c

#Correlation plot from numeric variables
ggcorrplot(cor(musiclognum), hc.order = TRUE, outline.col = "white",lab = TRUE, title = "Correlation Pl
```



The correlation plot helps showcase the relationship between numeric variables. instrumentalness variable has the highest correlation(.41) with the Target variable, followed by danceability.

Logistic Regression Model

```
musiclog$decade = as.factor(musiclog$decade)#converting decade variable to factor
set.seed(123)#used to randomize the records in the dataset

train_set <- musiclog %>% filter(decade != "0" & decade != "10" )#training set obtained by filtering on
train_set <-train_set[,-c(17)]

test_set <- filter(musiclog, decade == "0" | decade == "10")#testing set obtained by filtering on data
test_set <-test_set[,-c(17)]
```

```
logitreg <- glm(target~.,data = train_set, family = "binomial")#Creating a logistic regression model ba

#Model summary
summary(logitreg)
```

```
##
## Call:
## glm(formula = target ~ ., family = "binomial", data = train_set)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0413  -1.0357   0.2244   0.9346   2.9066
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)       4.707e-01  1.511e-01   3.115 0.001840 **
## mode              3.440e-01  2.724e-02  12.630  < 2e-16 ***
## speechiness      -2.859e+00  1.703e-01 -16.788  < 2e-16 ***
## acousticness     -1.982e+00  4.588e-02 -43.198  < 2e-16 ***
## instrumentalness -3.700e+00  6.730e-02 -54.975  < 2e-16 ***
## liveness         -8.322e-01  7.063e-02 -11.783  < 2e-16 ***
## tempo            -1.600e-03  4.372e-04  -3.660 0.000253 ***
## duration_ms      -1.124e-07  2.782e-07  -0.404 0.686201
## time_signature    3.081e-01  3.194e-02   9.646  < 2e-16 ***
## chorus_hit       -2.887e-03  7.344e-04  -3.931 8.45e-05 ***
## sections         -8.534e-03  6.648e-03  -1.284 0.199213
## decade.L         -5.345e-01  3.223e-02 -16.582  < 2e-16 ***
## decade.Q          2.360e-01  2.829e-02   8.344  < 2e-16 ***
## decade.C         -5.442e-02  2.842e-02  -1.915 0.055510 .
## decade^4         -1.441e-01  2.785e-02  -5.175 2.28e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 48116  on 34707  degrees of freedom
## Residual deviance: 38976  on 34693  degrees of freedom
## AIC: 39006
##
## Number of Fisher Scoring iterations: 5
```

A logistic regression model was created based on all the independant variables and the dependant variable

Target(1/0).
From the model summary it is clear that at 95% confidence interval all the independendant variables except - valence,time_signature and duration_ms were statistically significant in determining the Target(1/0). This was confirmed as the variables had p-value less than 0.05 thus rejecting the null hypothesis that coefficient = 0.

By keeping the cutoff for the Target variable to be considered as hit to .6 , a confusion matrix was created. The confusion matrix shows the model has an accuracy of 74.56% which is greater than the No - information rate. The confusion matrix also shows the sensitivity(82.46%) and specificity(66.6%) for the model.

Variable Importance as Per T-Statistic

```
#Creating a data frame showing variable importance in decreasing order of importnace based on T-statist
imp <- as.data.frame(varImp(logitreg))
imp <- data.frame(Variable_names   = rownames(imp),overall_Importance_Tstat = imp$Overall)
imp[order(imp$overall,decreasing = T),]
```

```
##      Variable_names overall_Importance_Tstat
## 4  instrumentalness               54.9745962
## 3      acousticness               43.1984453
## 2       speechiness               16.7877656
## 11         decade.L               16.5818774
## 1              mode               12.6304288
## 5          liveness               11.7834954
## 8    time_signature                9.6462618
## 12         decade.Q                8.3440273
## 14         decade^4                5.1747971
## 9        chorus_hit                3.9313089
## 6             tempo                3.6597106
## 13         decade.C                1.9148641
## 10         sections                1.2837968
## 7       duration_ms                0.4040153
```

```
#machine learning model for predicting hits, or determining how a song would become a hit (XGB, logisti






#different formula potentially for first hit vs afterwards like lets say the song hit #40 would that ch




#after training models look for equivalent dataset to see if there is something for 2020 so far

#https://www.kaggle.com/theoverman/the-spotify-hit-predictor-dataset
#https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/
#https://towardsdatascience.com/song-popularity-predictor-1ef69735e380
#might need to scrape billboard top 100 to get the list of songs we need to test -> what about the othe

#https://github.com/manasreldin/Song-Popularity-Predictor/blob/master/Scrape_BB.ipynb -> datascrape bil
```

```
#https://github.com/manasreldin/Song-Popularity-Predictor/blob/master/demo.py

#https://github.com/manasreldin/Song-Popularity-Predictor/blob/master/SimpleFeatures.csv

#https://github.com/manasreldin/Song-Popularity-Predictor/blob/master/PredictHotBB.ipynb
```

As a result of there not being any significant data to suggest whether or not there is a a difference between tempo we test other aspects of the dataframe to see if there are any indictors to show separation between songs that are hits and songs that are not.